Fachhochschule Köln
Cologne University of Applied Sciences

VIAT

User's Guide

v0.3

Verfahren zur Identifikation und Abwehr von Telefon-SPAM
(Translation: *Method for the Identification and Blocking of Telephone-SPAM*)

Prof. Dr. Heiko Knospe
M.Sc. Dirk Lentzen
Dipl.-Ing.(FH) Gary Grutzek

Prof. Dr. Christoph Pörschmann
M.Sc. Julian Strobl
Dipl.-Ing.(FH) Bernhard Mainka

SPONSORED BY THE
Federal Ministry of Education and Research

Institute of Communications Engineering
Department of IT Security
November 27, 2012

Project Partners:

In cooperation with:

Acknowledgments:

A part of this work was carried out during an internship with Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE, Wachtberg, Germany. We would especially like to thank Frank Kurth for his support.

For more information visit: http://viat.fh-koeln.de/

Contents

1 Introduction **1**

2 VirtualBox Appliance: VIAT.ova **2**

2.1 Debian GNU/Linux 64 Bit 3

2.1.1 User Accounts 3

2.1.2 Network Configuration 3

2.1.3 Additional Packages 3

2.1.4 Boost Library v1.48 with Boost Log v1.1 4

2.1.5 PostgreSQL v8.4 Database 4

2.1.6 FrameWave 1.3.1 5

2.1.7 Asterisk A and B Communication Servers 5

2.1.8 Directories 6

2.2 Usage 7

2.2.1 A Basic Example 8

2.2.2 Clear The Blacklist 10

2.2.3 Create Your Own Call Scenarios 10

References **ii**

List of Figures

1 VIAT Testing Environment. 2

2 VIAT Database Scheme. 5

3 VIAT Testing Environment opened in terminal. 7

4 PostgreSQL administration tool opened in web browser. 8

5 Call flow from `Asterisk A` (left) to `Asterisk B` (right). 8

6 Partial output of `indexd.log` with a similarity of 40% between the two calls. 9

7 Information about blocked caller in `Asterisk A` (left). 9

List of Listings

1 Network configuration: /etc/network/interfaces. 4

2 Example configuration for callfiles: /home/viat/scenario/minimal/config-make-callfiles-demo.pl. 10

3 Example configuration for calls: /home/viat/scenario/minimal/config-make-calls-demo.pl. 11

1 Introduction

VIAT means Verfahren zur Identifikation und Abwehr von Telefon-SPAM which translates to *Method for the Identification and Blocking of Telephone-SPAM*. VIAT is a research project (period of time: July 2009 - December 2012) of the Cologne University of Applied Sciences (project management) in cooperation with the **TU Braunschweig**, the **Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE** and the companies **IPTEGO GmbH** and **Sirrix AG**. The project is significantly financed by the **Federal Ministry of Education and Research** and supported by the involved companies.

Telephone-SPAM is characterized by bulk unsolicited calls. The SPAMer attempts to initiate a voice session and relays a prerecorded message if the callee answers. The prevalent Voice over IP (VoIP) technology provides convenient tools and low-priced possibilities to place a large number of SPAM calls.

This document is organized as follows: In the next section we describe our released VirtualBox Appliance which is already configured and with which you can easily use and test our prototype. You can even take your own audio material and create your own call scenarios. After that the papers that we have written, are listed.

2 VirtualBox Appliance: VIAT.ova

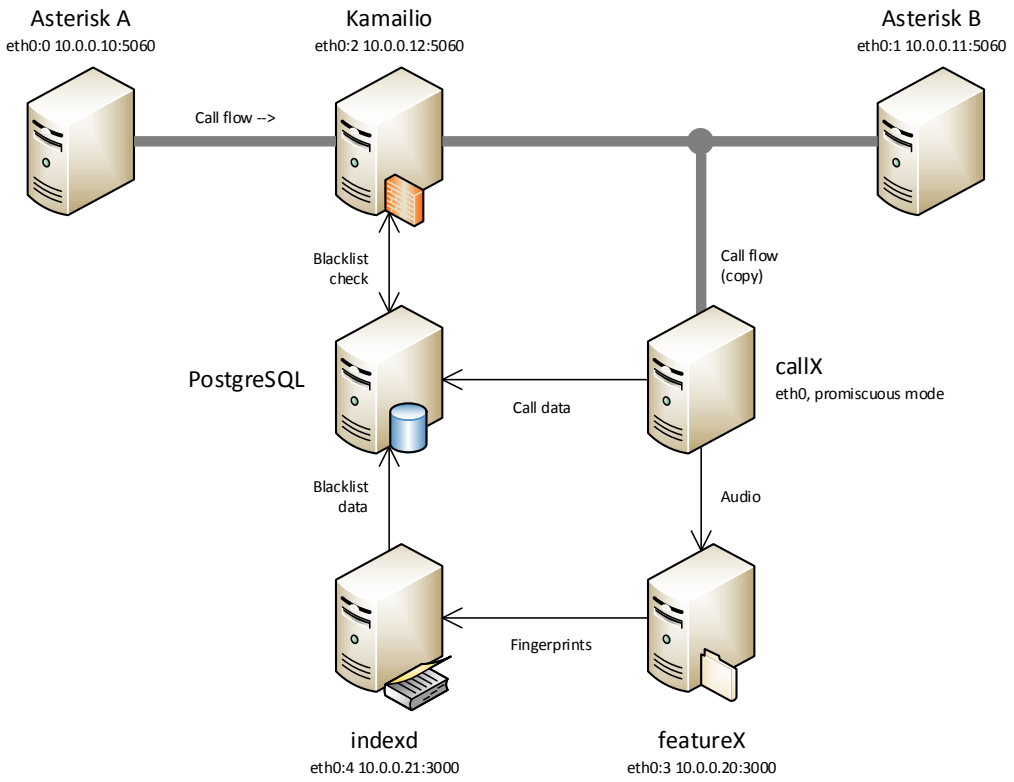

Figure 1: VIAT Testing Environment.

In figure 1 you can see the VIAT testing environment. There are two `Asterisk` communication servers. One for call generation called `Asterisk A` and one for call termination called `Asterisk B`. The SIP Express Router (SER) `Kamailio` transfers the calls between these servers, if the caller is not on the `Blacklist`.

From a copy of the network stream the audio data is extracted[1] (`callX`) and the audio fingerprint is computed (`featureX`). After that the fingerprints are compared against all previous fingerprints (`indexd`) and the information about possible equal or similar audio data is stored in the `PostgreSQL` database. With this information the `Blacklist` in the database is filled.

[1] Not yet integrated. The call extraction is handled by a perl-script which monitors the folder `/var/spool/asterisk/monitor/`. It writes the call metadata to the database and passes the audio material to featureX. The script is at `~/script/vmfd.pl`.

2.1 Debian GNU/Linux 64 Bit

The virtual machine is a Debian GNU/Linux 64 Bit system. In the following we describe the changes we have made to a fresh `debian-6.0.5` installation.

Security updates have been installed on November 2, 2012.

2.1.1 User Accounts

We have two users. On the hand the standard superuser `root` and an additional user `viat`, which should be taken to control the virtual machine and our system. The password for both accounts is: `viat`.

2.1.2 Network Configuration

We modified the network configuration to reach the scenario described in figure 1. You can see the changes in listing 1.

2.1.3 Additional Packages

Some additional packages have been installed:

```
openssh-server postgresql libpq5 libpq-dev asterisk
libdatetime-format-iso8601-perl
```

The `openssh-server` is for an easy connection e.g. via terminal to the virtual machine. The packages `postgresql`, `libpq5` and `libpq-dev` are for the PostgreSQL database, where the last one is needed for compiling our sources. The last two packages `asterisk` and `libdatetime-format-iso8601-perl` are for the open source telephony software `Asterisk` (see section 2.1.7).

Listing 1: Network configuration: /etc/network/interfaces.

```
allow-hotplug eth0
iface eth0 inet dhcp

auto eth0:0
iface eth0:0 inet static
        address 10.0.0.10
        netmask 255.255.255.0

auto eth0:1
iface eth0:1 inet static
        address 10.0.0.11
        netmask 255.255.255.0

auto eth0:2
iface eth0:2 inet static
        address 10.0.0.12
        netmask 255.255.255.0

auto eth0:3
iface eth0:3 inet static
        address 10.0.0.20
        netmask 255.255.255.0

auto eth0:4
iface eth0:4 inet static
        address 10.0.0.21
        netmask 255.255.255.0
```

2.1.4 Boost Library v1.48 with Boost Log v1.1

The libraries are stored in `/usr/lib/` and have the format `libboost_*.{a|so}`. The header files are in a new directory in `/usr/include/boost`.

2.1.5 PostgreSQL v8.4 Database

For further information about the installation process, the created database, tables and users see the `Developer's Guide`. In figure 2 you can see which information is stored in the database.

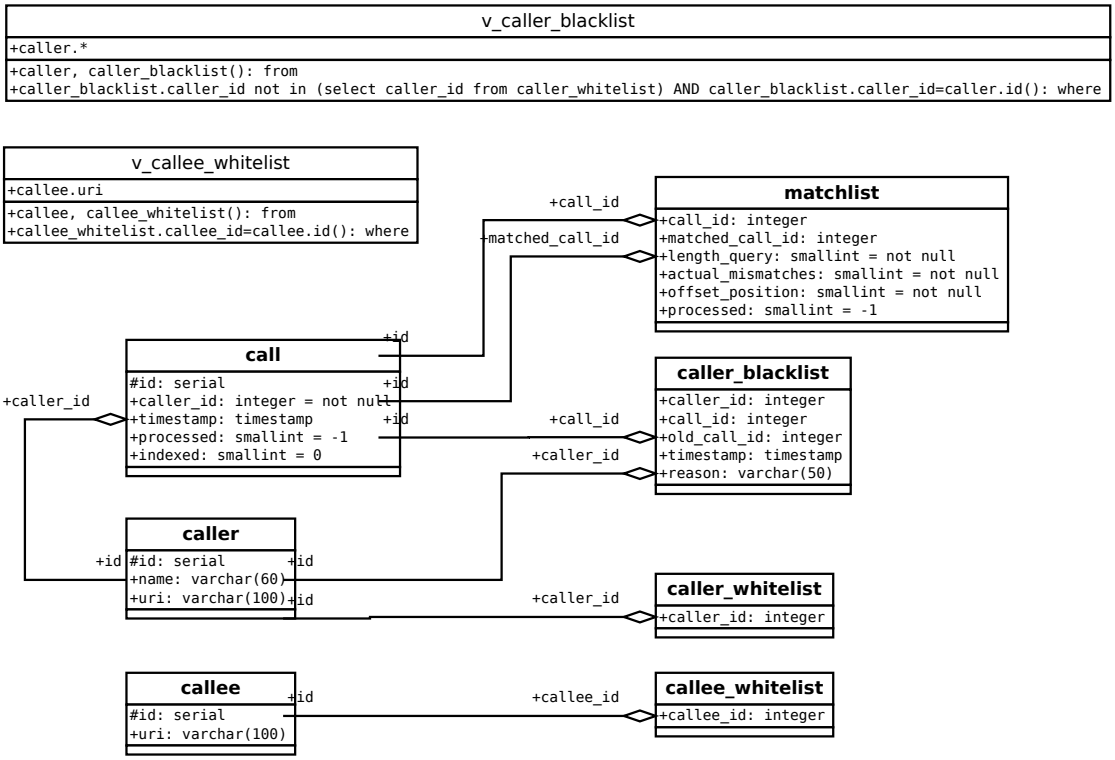

Figure 2: VIAT Database Scheme.

2.1.6 FrameWave 1.3.1

> *Framewave is a free and open-source collection of popular image and signal processing routines designed to accelerate application development, debugging, multi-threading and optimization on x86-class processor platforms.*[2]

The shared libraries object can be found at `/usr/lib/` starting with `fw*`. The libraries are needed by `featureX`.

2.1.7 Asterisk A and B Communication Servers

We have two `Asterisk` servers in our testing environment. One for call generation `Asterisk A` and one for call termination `Asterisk B` (see figure 1), so that we can simulate a full call flow.

[2] http://framewave.sourceforge.net/

Each `Asterisk` server needs its own network interface, so that we can passively extract the SIP and RTP data (`callx`). Therefore we created two virtual network interfaces. The interface `eth0:0` is for `Asterisk A` and the interface `eth0:1` is for `Asterisk B` (see figure 1 and listing 1).

To start two instances of the `asterisk` software, we have two modified start scripts `/etc/init.d/asta` and `/etc/init.d/astb`. Furthermore we removed the default start script and added our two modified ones. This way you always have the `Asterisk A` and `Asterisk B` running at startup.

For every instance we have new directories for the running directory:

```
drwxr-xr-x 2 viat viat /var/run/asta
drwxr-xr-x 2 viat viat /var/run/astb
```

and the configuration directory:

```
drwxr-xr-x 2 viat viat /etc/asta
drwxr-xr-x 2 viat viat /etc/astb
```

Finally we created two new executables:

```
/usr/sbin/asta
/usr/sbin/astb
```

and you can connect to the specific instance by passing the `-r` argument!

2.1.8 Directories

All directories are owned by `viat`. The configuration files of our software can be found in `/etc/viat/`, logging output is in `/var/log/viat/` and pid-files are in `/var/run/viat/`. The configuration files of the `Asterisk` instances can be found in `/etc/asta/` and `/etc/astb/`, logging output in `/var/log/asta/` and `/var/log/astb/` and the pid-files in `/var/run/asta/` and `/var/run/astb/`. The transferred audio material (.wav) can be found in the directories `/var/spool/asta/` and `/var/spool/astb/`.

2.2 Usage

When starting the image two things will automatically open. On the one hand a couple of terminals with which you can control our environment (see figure 3).

In the top section you can see the console of `Asterisk A` and `Asterisk B`. Theses are just for output. Later we will see our call flow and blocked calls here. In the middle section is the output of `indexd`. If you are interested in the search algorithm, you will get some information in [1]. The bottom section is for actually control our system. We have prepared some scripts for you to get our system easily running.

One the other hand you see the `PostgreSQL` administration tool opened in a browser (see figure 4). Username and password are both `viat`.

In the beginning the database is empty. After running our system a little while interesting tables are `matchlist`, where you can see the search output about similar calls; the `call` table with metadata about all calls and of course the `caller_blacklist`, where the SER `Kamailio` gets its information about blocking certain callers.

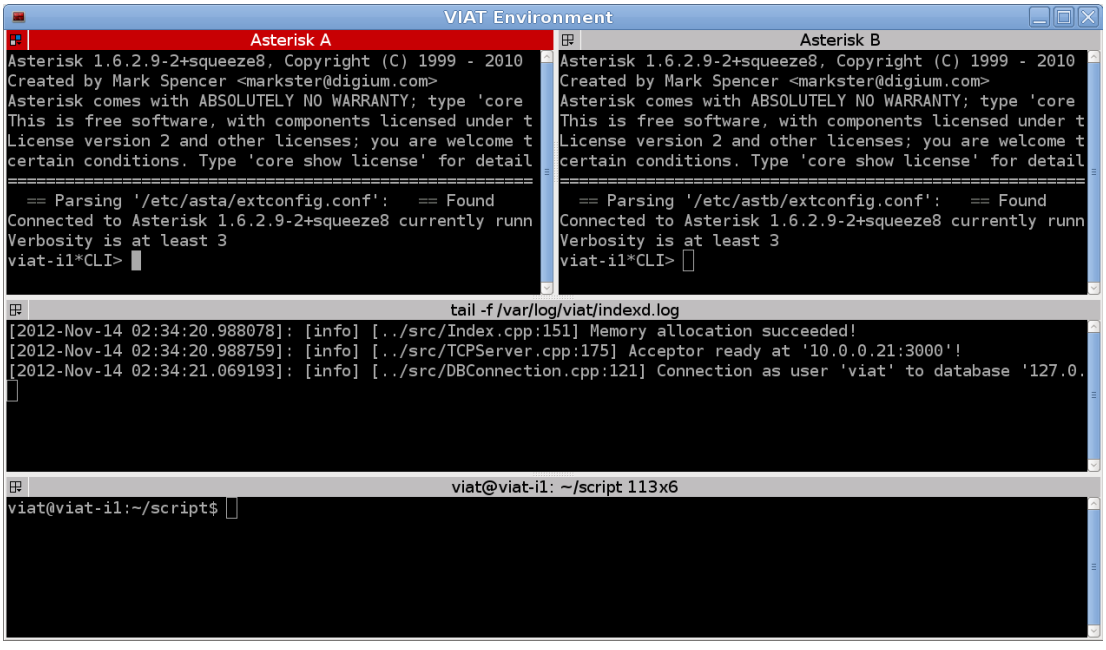
VIAT Environment
Asterisk A
Asterisk 1.6.2.9-2+squeeze8, Copyright (C) 1999 - 2010
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core
This is free software, with components licensed under t
License version 2 and other licenses; you are welcome t
certain conditions. Type 'core show license' for detail
== Parsing '/etc/asta/extconfig.conf': == Found
Connected to Asterisk 1.6.2.9-2+squeeze8 currently runn
Verbosity is at least 3
viat-i1*CLI>
Asterisk B
Asterisk 1.6.2.9-2+squeeze8, Copyright (C) 1999 - 2010
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core
This is free software, with components licensed under t
License version 2 and other licenses; you are welcome t
certain conditions. Type 'core show license' for detail
== Parsing '/etc/astb/extconfig.conf': == Found
Connected to Asterisk 1.6.2.9-2+squeeze8 currently runn
Verbosity is at least 3
viat-i1*CLI>
tail -f /var/log/viat/indexd.log
[2012-Nov-14 02:34:20.988078]: [info] [../src/Index.cpp:151] Memory allocation succeeded!
[2012-Nov-14 02:34:20.988759]: [info] [../src/TCPServer.cpp:175] Acceptor ready at '10.0.0.21:3000'!
[2012-Nov-14 02:34:21.069193]: [info] [../src/DBConnection.cpp:121] Connection as user 'viat' to database '127.0.
viat@viat-i1: ~/script 113x6
viat@viat-i1:~/script$

Figure 3: VIAT Testing Environment opened in terminal.

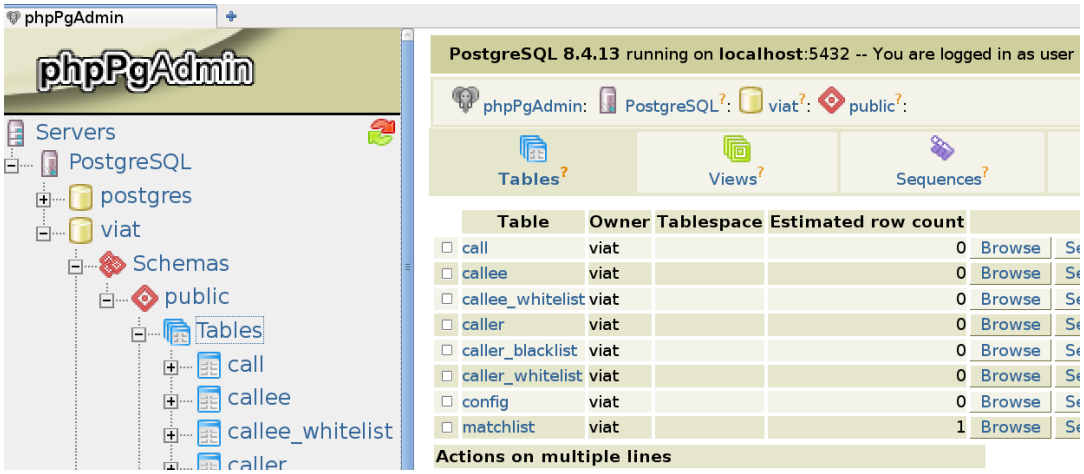

Figure 4: PostgreSQL administration tool opened in web browser.

2.2.1 A Basic Example

To get you started very fast, we provide an example called `minimal`. Just run from the `~/script` directory and see what happens:

```
./make-calls.sh /home/viat/scenario/minimal demo
```

Two calls are transmitted. In the top section you see the call flow from `Asterisk A` to `Asterisk B` (see figure 5). When the calls are finished the search is performed and we

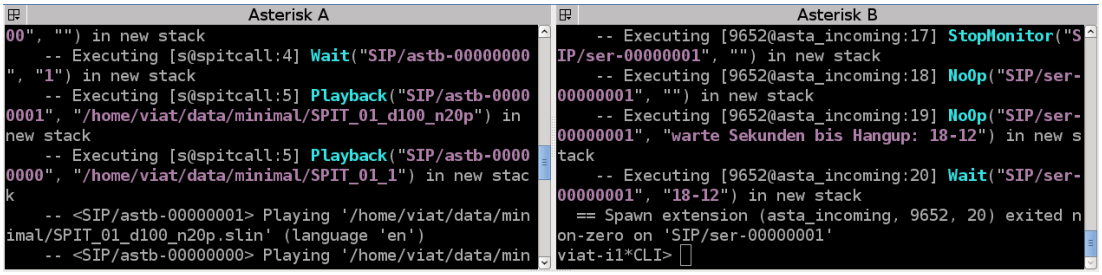
Asterisk A
00", "") in new stack
-- Executing [s@spitcall:4] Wait("SIP/astb-00000000", "1") in new stack
-- Executing [s@spitcall:5] Playback("SIP/astb-00000001", "/home/viat/data/minimal/SPIT_01_d100_n20p") in new stack
-- Executing [s@spitcall:5] Playback("SIP/astb-00000000", "/home/viat/data/minimal/SPIT_01_1") in new stack
-- <SIP/astb-00000001> Playing '/home/viat/data/minimal/SPIT_01_d100_n20p.slin' (language 'en')
-- <SIP/astb-00000000> Playing '/home/viat/data/min
Asterisk B
-- Executing [9652@asta_incoming:17] StopMonitor("SIP/ser-00000001", "") in new stack
-- Executing [9652@asta_incoming:18] NoOp("SIP/ser-00000001", "") in new stack
-- Executing [9652@asta_incoming:19] NoOp("SIP/ser-00000001", "warte Sekunden bis Hangup: 18-12") in new stack
-- Executing [9652@asta_incoming:20] Wait("SIP/ser-00000001", "18-12") in new stack
== Spawn extension (asta_incoming, 9652, 20) exited non-zero on 'SIP/ser-00000001'
viat-i1*CLI>

Figure 5: Call flow from `Asterisk A` (left) to `Asterisk B` (right).

get a similarity between these two calls (see figure 6). The result says that the `actual mismatches` are 60 and since the fingerprint length of each call is 100, the two calls have 40 features in the right distance in common. We can also see this information in the database table `matchlist`. With this information the table `caller_blacklist` is filled, since we require a similarity of at least 15% (see `/etc/viat/mld.conf`).

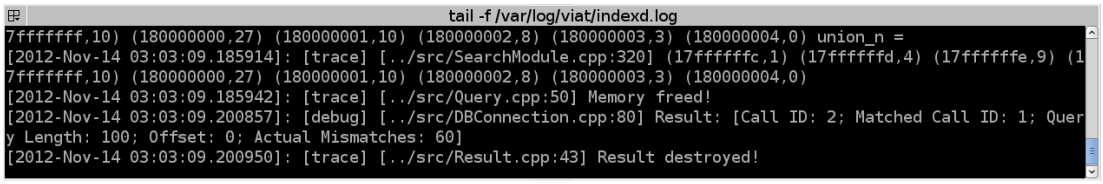
tail -f /var/log/viat/indexd.log
7fffffff,10) (180000000,27) (180000001,10) (180000002,8) (180000003,3) (180000004,0) union_n =
[2012-Nov-14 03:03:09.185914]: [trace] [../src/SearchModule.cpp:320] (17ffffffc,1) (17ffffffd,4) (17ffffffe,9) (1
7fffffff,10) (180000000,27) (180000001,10) (180000002,8) (180000003,3) (180000004,0)
[2012-Nov-14 03:03:09.185942]: [trace] [../src/Query.cpp:50] Memory freed!
[2012-Nov-14 03:03:09.200857]: [debug] [../src/DBConnection.cpp:80] Result: [Call ID: 2; Matched Call ID: 1; Quer
y Length: 100; Offset: 0; Actual Mismatches: 60]
[2012-Nov-14 03:03:09.200950]: [trace] [../src/Result.cpp:43] Result destroyed!

Figure 6: Partial output of `indexd.log` with a similarity of 40% between the two calls.

Note: Although the audio material came from different callers, our system recognizes the similarity and is able to block both callers!

We can now try to make the two calls again, but now we are blocked (see figure 7). Even

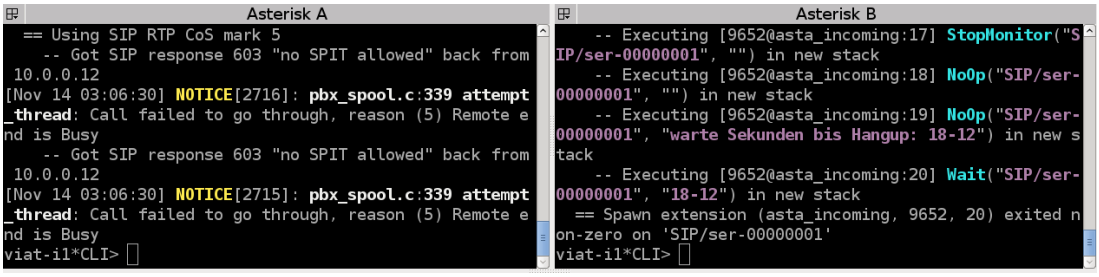
Asterisk A
== Using SIP RTP CoS mark 5
-- Got SIP response 603 "no SPIT allowed" back from
10.0.0.12
[Nov 14 03:06:30] NOTICE[2716]: pbx_spool.c:339 attempt
_thread: Call failed to go through, reason (5) Remote e
nd is Busy
-- Got SIP response 603 "no SPIT allowed" back from
10.0.0.12
[Nov 14 03:06:30] NOTICE[2715]: pbx_spool.c:339 attempt
_thread: Call failed to go through, reason (5) Remote e
nd is Busy
viat-i1*CLI>
Asterisk B
-- Executing [9652@asta_incoming:17] StopMonitor("S
IP/ser-00000001", "") in new stack
-- Executing [9652@asta_incoming:18] NoOp("SIP/ser-
00000001", "") in new stack
-- Executing [9652@asta_incoming:19] NoOp("SIP/ser-
00000001", "warte Sekunden bis Hangup: 18-12") in new s
tack
-- Executing [9652@asta_incoming:20] Wait("SIP/ser-
00000001", "18-12") in new stack
== Spawn extension (asta_incoming, 9652, 20) exited n
on-zero on 'SIP/ser-00000001'
viat-i1*CLI>

Figure 7: Information about blocked caller in `Asterisk A` (left).

if we would take other call files we wouldn't get through.

Play The Audio Files Play the audio files and observe that they only differ in noise levels and a little delay of 100 ms.

```
totem /home/viat/data/minimal/SPIT_01_1.wav
totem /home/viat/data/minimal/SPIT_01_d100_n20p.wav
```

Monitor The Packets We installed `Wireshark` for you. Just start it as root, e.g.

```
gksudo wireshark
```

and monitor the loopback device `lo`. To get the VoIP traffic only, set the filter:

```
sip or rtp
```

and hit `Apply`.

2.2.2 Clear The Blacklist

If you just want to clear the blacklist but not restart our system, you can use the `PostgreSQL Administration Tool` (see figure 4). Just click `caller_blacklist` and then `Empty`. Done!

2.2.3 Create Your Own Call Scenarios

Create Call Files First of all we need a configuration file similar to the example you see in listing 2.

Listing 2: Example configuration for callfiles: /home/viat/scenario/minimal/config-make-callfiles-demo.pl.

```
1 inpath =>         '/home/viat/data/minimal',
2 outpath =>        '/home/viat/scenario/minimal/callfiles',
3 callfilename =>   'minimal',
4 template =>       '/home/viat/scenario/template.call',
5 media_files =>    [ 'wav', 'mp3', 'gsm' ],
6 mincaller =>      2211000,
7 maxcaller =>      2211010,
8 mincallee =>      9100,
9 maxcallee =>      9999,
```

Finally we change in the ∼`/script` directory and run the following command:

```
./create-callfiles.pl --configfile=\
/home/viat/scenario/minimal/config-make-callfiles-demo.pl
```

Starting Calls We need a configuration file to simulate our call flow fitting our needs. You can see an example in listing 3.

Listing 3: Example configuration for calls: /home/viat/scenario/minimal/config-make-calls-demo.pl.

```
inpath =>               '/home/viat/scenario/minimal/callfiles/',
tmppath =>              '/home/viat/scenario/minimal/tmp',
outpath =>              '/var/spool/asta/outgoing',
# 0=play each file just once, 1=replays are possible
mitzuruecklegen =>      0,
# file extension of callfiles
call_files =>           'call',
# start time of simulation
starttime =>            '2010-03-28T11:00:00',
# seconds of simulated time slice
ticktime =>             5,
# time of time slice reality
ticklength =>           5,
# maximum number of calls per tick
#call_max =>            15,
# minimal number of calls per tick
#call_min =>            5,
# level of debug output
verbose =>              3,
hour_loads =>           [
                        1,1,1,1,1,1,  # 00:00-05:59
                        1,1,1,1,1,1,  # 06:00-11:59
                        1,1,1,1,1,1,  # 12:00-17:59
                        1,1,1,1,1,1   # 18:00-24:00
                        ],
# number of calls per tick in busyhour
busyhour_calls =>       2,
# Influence of randomness? 10 equals to +- 5
rand_factor    =>       0,
# play callfiles randomly? 0=play callfiles in order, 1=play randomly
random         =>       0,
```

From the ~/`script` directory just run:

```
./make-calls.sh /home/viat/scenario/minimal demo
```

References

[1] J. Strobl, G. Grutzek, B. Mainka, and H. Knospe, "An Efficient Search Method for the Content-Based Identification of Telephone-SPAM," in *IEEE International Conference on Communications (ICC)*, pp. 2656–2660, june 2012. 7

[2] G. Grutzek, J. Strobl, B. Mainka, F. Kurth, C. Poerschmann, and H. Knospe, "A Perceptual Hash for the Identification of Telephone Speech," in *2012 ITG Fachtagung Sprachkommunikation*, september 2012.

[3] D. Lentzen, G. Grutzek, H. Knospe, and C. Poerschmann, "Content-Based Detection and Prevention of Spam over IP Telephony - System Design, Prototype and First Results," in *IEEE International Conference on Communications (ICC)*, pp. 251–252, june 2011.

[4] J. Strobl, F. Kurth, G. Grutzek, and H. Knospe, "Effiziente Identifikation von Telefon-Spam," in *Fortschritte der Akustik - DAGA 2011, DEGA e.V.*, pp. 251–252, 2011.

[5] G. Grutzek, C. Poerschmann, and H. Knospe, "Vergleich spektraler Merkmale zur Identifikation von Telefon SPAM," in *Fortschritte der Akustik - DAGA 2010, DEGA e.V.*, pp. 243–244, 2010.

[6] H. Knospe and C. Poerschmann, "Ein neues Verfahren zur Identifikation und Abwehr von Telefon-SPAM," in *Scientific Reports of the Cologne University of Applied Sciences, Proceedings des XXI. Deutsch-Polnischen Seminars*, pp. 49–53, 2009.

[7] C. Poerschmann and H. Knospe, "Spectral Analysis of Audio Signals for the Identification of Spam Over IP Telephony," in *Proceedings of the NAG/DAGA 2009, DEGA e.V*, pp. 1027–1029, 2009.

[8] C. Poerschmann and H. Knospe, "Analyse spektraler Parameter des Audiosignals zur Identifikation und Abwehr von Telefon-SPAM," in *Sicherheit 2008, Lecture Notes in Informatics, Proceedings Sicherheit 2008, Gesellschaft für Informatik*, vol. P-128, pp. 551–555, 2008.

[9] C. Poerschmann and H. Knospe, "Analysis of Spectral Parameters of Audio Signals for the Identification of Spam Over IP Telephony," in *The Fifth Conference on Email and Anti-Spam*, pp. 551–555, 2008.